Eine Übersicht über unterbrechende Ausführung

Ein Vergleich von Ansätzen zum Verhindern von durch Unterbrechungeingeführten Fehlern

5. August 2020

Colin A. Voigt

Friedrich-Alexander-Universität Erlangen-Nürnberg

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Motivation

Hyundai recalls 430,000 Elantra models for suddenly catching fire

Even when the cars are not running, a short circuit could cause a fire in the engine compartment.

Sean Szymkowski Feb. 12, 2020 12:13 p.m. PT

LISTEN · 01:03

Dog dies in Taihape blaze, charging battery to blame

Maxine Jacobs · 16:44, Jul 20 2020

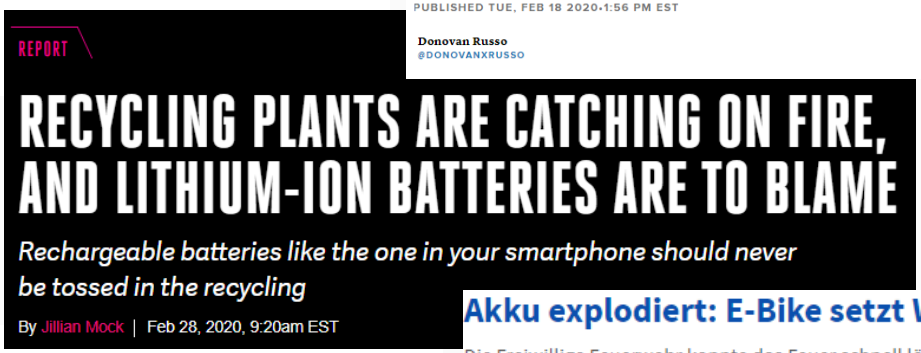
TRANSPORTATION

Electric Porsche Taycan catches fire while parked overnight in garage, company confirms

PUBLISHED TUE, FEB 18 2020•1:56 PM EST

Donovan Russo
@DONOVANXRUSSO

SHARE



REPORT

RECYCLING PLANTS ARE CATCHING ON FIRE, AND LITHIUM-ION BATTERIES ARE TO BLAME

Rechargeable batteries like the one in your smartphone should never be tossed in the recycling

By Jillian Mock | Feb 28, 2020, 9:20am EST

Verbot in Flugzeugen

US-Behörde stuft Samsung-Handy als "gefährliches Material" ein

Das Pannenhandy von Samsung wird an US-Flughäfen künftig eingestuft wie etwa Dynamit - Passagiere dürfen es nicht mit an Bord nehmen. Anderenfalls drohen Strafen.

15.10.2016, 18.13 Uhr

Akku explodiert: E-Bike setzt Wohnzimmer in Brand

Die Freiwillige Feuerwehr konnte das Feuer schnell löschen - **11.04.2020 14:51 Uhr**

ERLANGEN - Böse Osterüberraschung für eine Familie im Erlanger Röthelheimpark: Weil der Akku eines Elektro-Fahrrades beim Laden explodierte, ist das gesamte Wohnzimmer ausgebrannt.

Quellen: Hyundai Rückruf, Hund in Taihape, Recycling, Porsche Tycan, Samsung Note 7, Erlangen E-Bike

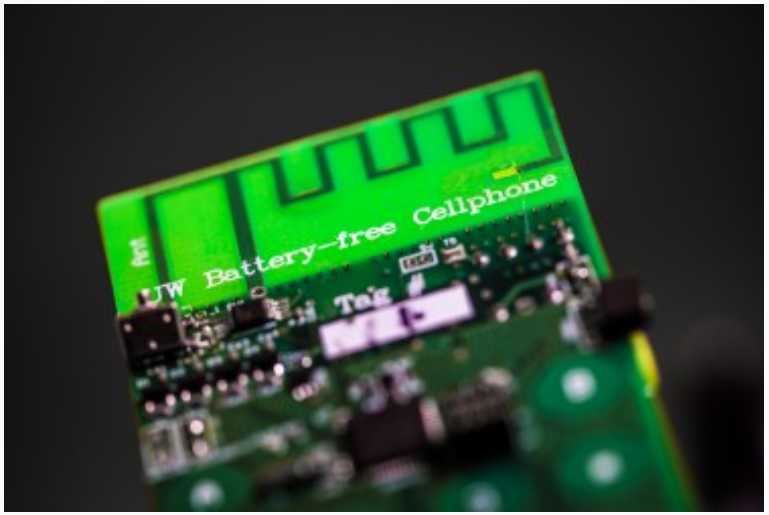
Motivation

Idee: Keine Batterien!

Problem: Ohne Strom rechnet kein Computer.

Lösung: Selber Energie gewinnen

Werbebild TacNet Rheinmetall



erstes Handy ohne Akku

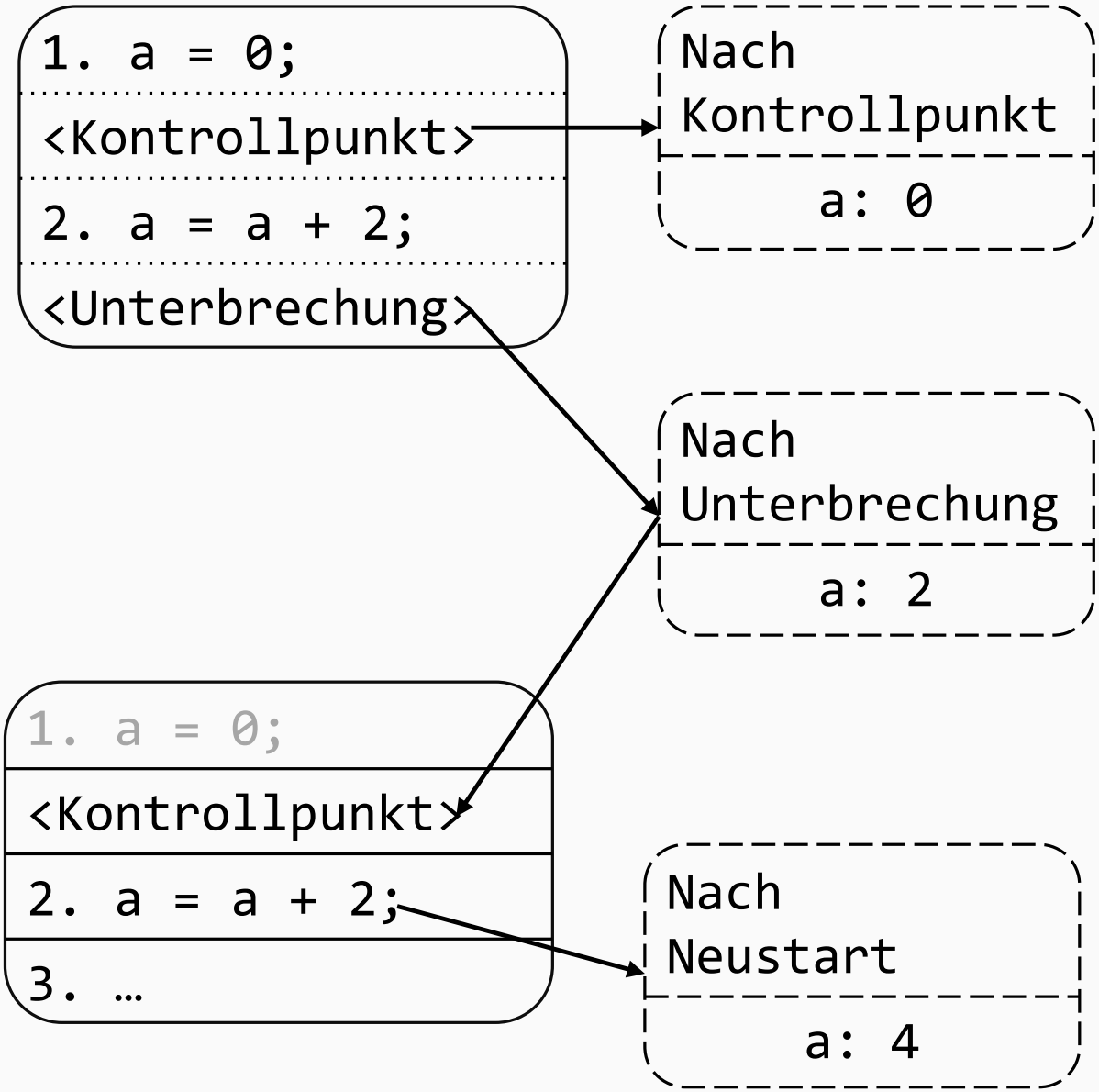
Inhalt

Fehler

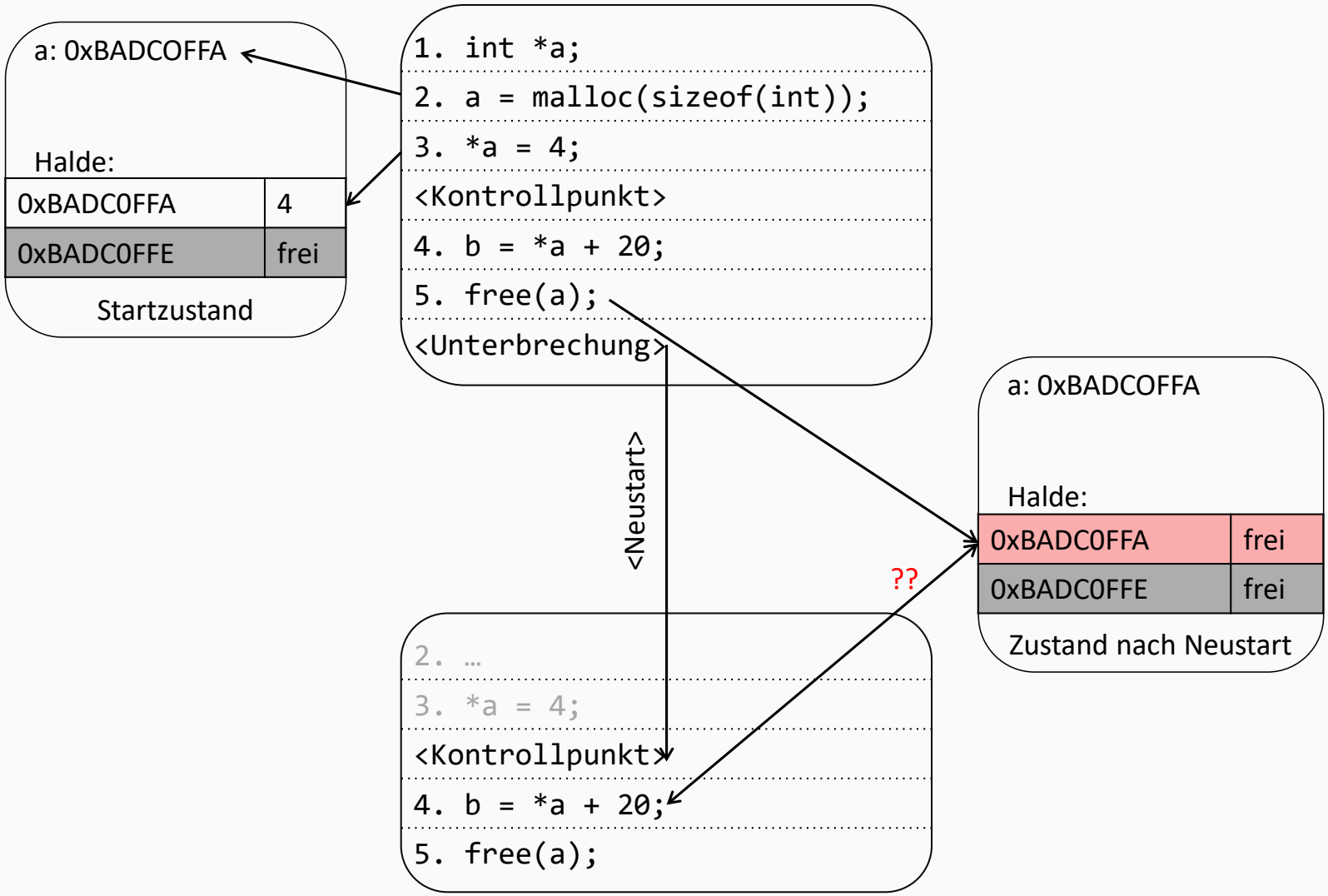
Übersicht: Fehler

1. Fehler
    - Datenzugrifffehler
    - Speicherabbild-Fehler
    - Fehler bei der Aktivierungsaufzeichnung
2. Ansätze
3. Vergleich
4. Fazit

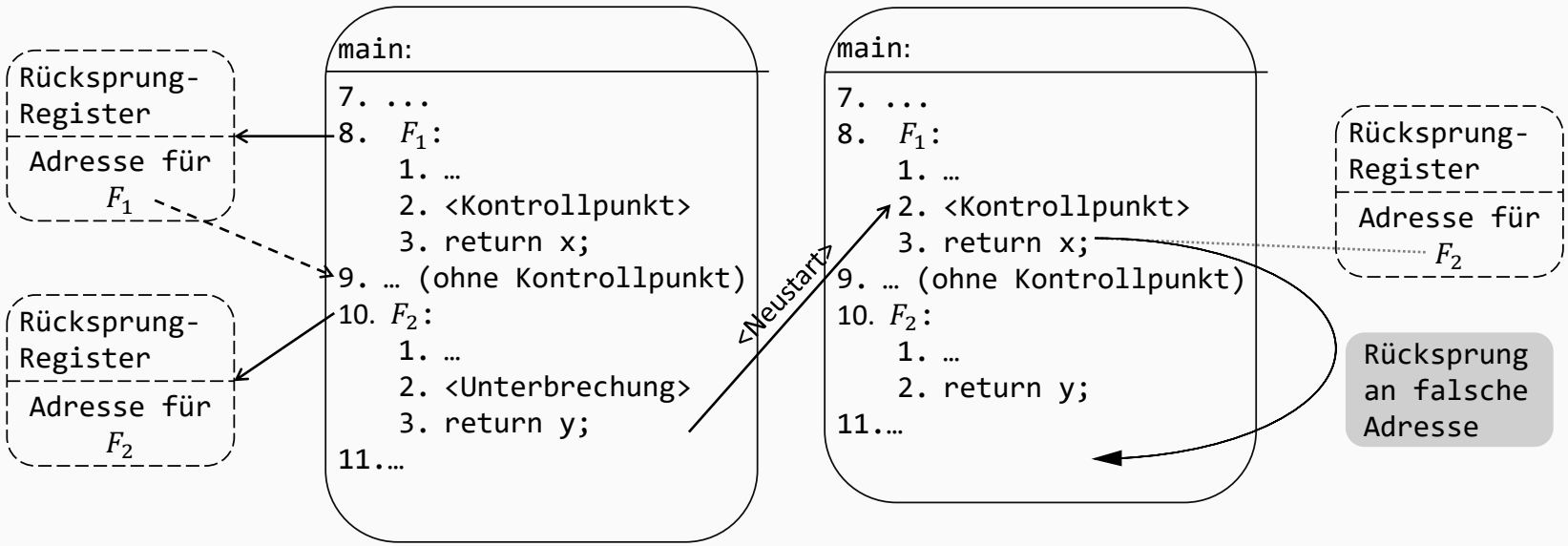
Datenzugrifffehler



Speicherabbild-Fehler



Fehler bei der Aktivierungsaufzeichnung



Ansätze

Übersicht: Ansätze

1. Fehler

2. Ansätze
   - DINO
   - Alpaca

3. Vergleich

4. Fazit

DINO

- **DINO** (**D**eath **I**s **N**ot an **O**ption)
- von Lucia und Ransford 2015 [LR15]
- keine Hardwareunterstützung benötigt

DINO: Programmiermodell

- Erweitert C Semantik um Aufgaben
  - Aufgaben immer atomar
  - Aufgaben als Pfad zwischen Aufgabengrenzen

DINO: Programmiermodell

- Erweitert C Semantik um Aufgaben
  - Aufgaben immer atomar
  - Aufgaben als Pfad zwischen Aufgabengrenzen
- Aufgabengrenzen statisch vom Entwickler deklariert
- Nach Unterbrechung wird an Aufgabengrenze neu gestartet

DINO: Ausführungsmodell

- Zwei Mechanismen werden verwendet
- Datenversionsverlauf
  - persistente Daten auf Stapel speichern

DINO: Ausführungsmodell

- Zwei Mechanismen werden verwendet
- Datenversionsverlauf
  - persistente Daten auf Stapel speichern
- Kontrollpunkte
  - jede Aufgabengrenze ist ein Kontrollpunkt
  - an Kontrollpunkt wird Stapel gesichert
  - Richtigkeit durch Prüfzahl prüfen
  - Kontrollpunkte sind doppelt gepuffert

Alpaca

- Von Maeng et al. 2017 [MCL17]

Alpaca

- Von Maeng et al. 2017 [MCL17]
- Basiert auf Aufgaben

Alpaca

- Von Maeng et al. 2017 [MCL17]
- Basiert auf Aufgaben
- Aufgabe darf nicht länger brauchen als Energie vorhanden ist

Alpaca

- Von Maeng et al. 2017 [MCL17]
- Basiert auf Aufgaben
- Aufgabe darf nicht länger brauchen als Energie vorhanden ist
- Keine Hardwareunterstützung

Alpaca

- Von Maeng et al. 2017 [MCL17]
- Basiert auf Aufgaben
- Aufgabe darf nicht länger brauchen als Energie vorhanden ist
- Keine Hardwareunterstützung
- Keine Kontrollpunkte

Alpaca

- Von Maeng et al. 2017 [MCL17]
- Basiert auf Aufgaben
- Aufgabe darf nicht länger brauchen als Energie vorhanden ist
- Keine Hardwareunterstützung
- Keine Kontrollpunkte
- Nutzt privatisierendes Speichermodell

Speichermodell

- unterscheidet zwischen:
- Aufgabenbezogene Daten
  - Nur in Aufgabe sichtbar
  - In Aufgabe deklariert
  - Im volatilen Speicher

Speichermodell

- unterscheidet zwischen:
- Aufgabenbezogene Daten
  - Nur in Aufgabe sichtbar
  - In Aufgabe deklariert
  - Im volatilen Speicher
- Aufgabengeteilte Daten
  - Sichtbar in allen Aufgaben
  - Global deklariert
  - Im persistenten Speicher
  - Während der Ausführung privatisiert

Privatisierung

- Nur verwendete, aufgabengeteilte Daten
- Statt kompletten Feld, Feldelemente

Privatisierung

- Nur verwendete, aufgabengeteilte Daten
- Statt kompletten Feld, Feldelemente
- statischer Privatisierungspuffer

Privatisierung

- Nur verwendete, aufgabengeteilte Daten
- Statt kompletten Feld, Feldelemente
- statischer Privatisierungspuffer
- Am Ende der Aufgabe werden Änderungen nach außen sichtbar

Nach außern sichtbar machen

Prozess in zwei Schritte unterteilt:

Nach außern sichtbar machen

Prozess in zwei Schritte unterteilt:

1. Vorübergeben
    - Es wird Liste, die alle neuen Werte enthält
    - Feldelemente beim ersten Schreiben
    - Skalare vor Übergeben

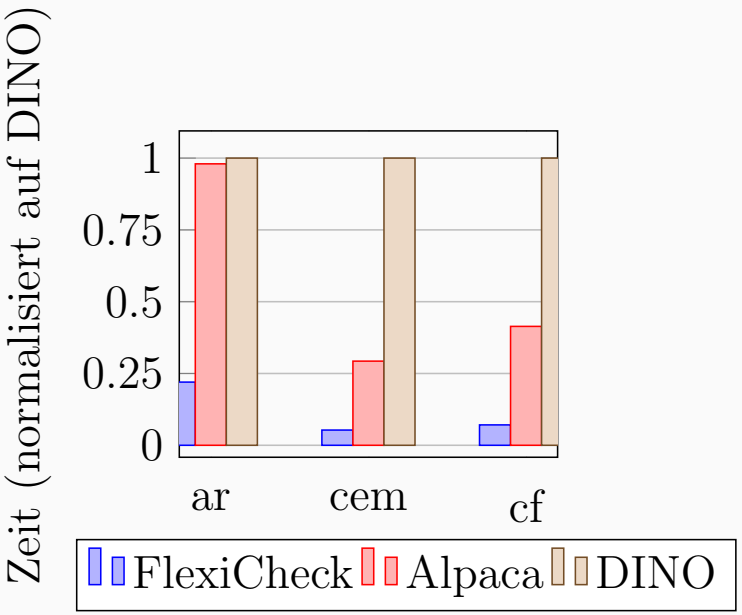
Nach außern sichtbar machen

Prozess in zwei Schritte unterteilt:

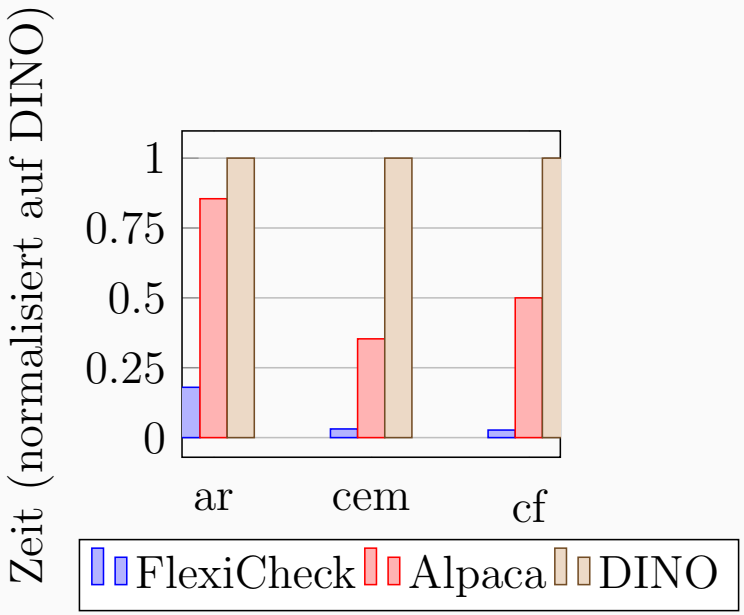
1. Vorübergeben
   - Es wird Liste, die alle neuen Werte enthält
   - Feldelemente beim ersten Schreiben
   - Skalare vor Übergeben
2. Übergeben
   - Markiert, dass Aufgabe vollständig durchlaufen ist
   - Setzt Markierungsbit, dass Vollständigkeit garantiert

Vergleich

Laufzeit: Die Messungen

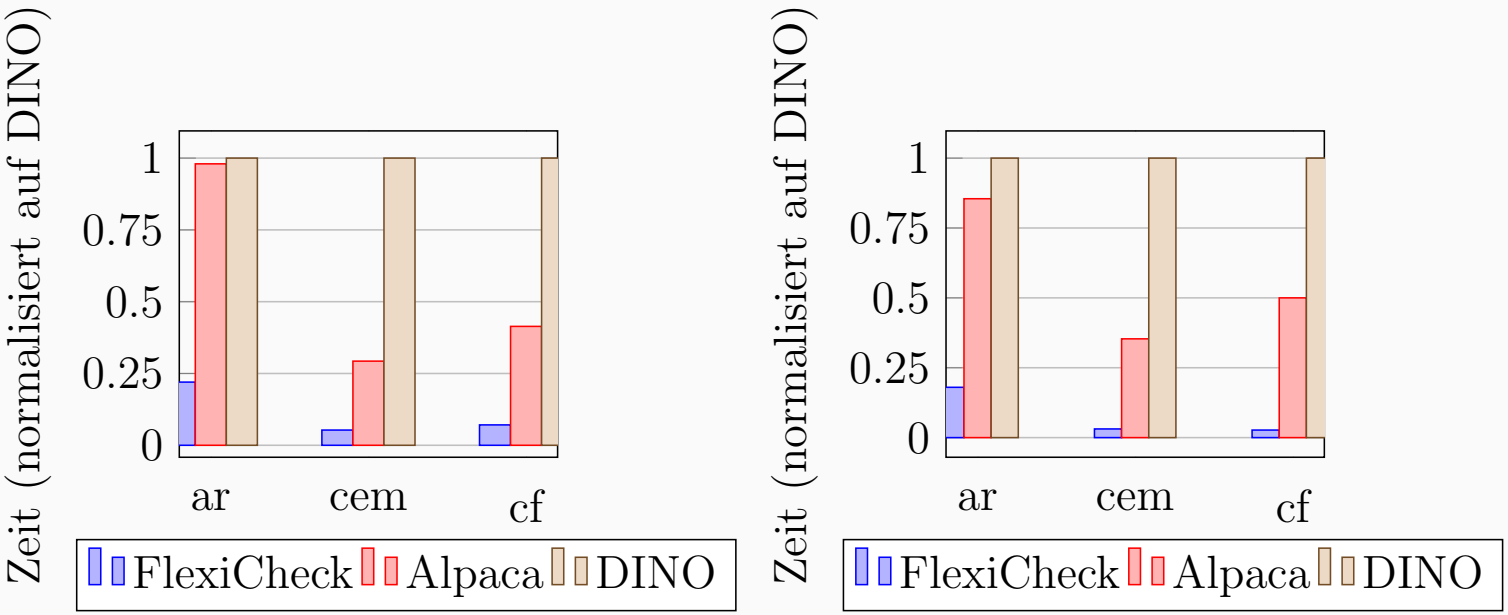


(a) bei kontinuierlicher Energie



(b) bei variabler Energie

Laufzeit: Die Messungen



(a) bei kontinuierlicher Energie

(b) bei variabler Energie

- Alpaca bis zu 15mal schneller
- FlexiCheck bis zu 20mal schneller

Entwickleraufwand

- Alpaca
  - benötigt längeren Kode

Entwickleraufwand

- Alpaca
  - benötigt längeren Kode und
  - mehr Schlüsselwörter als DINO

Entwickleraufwand

- Alpaca
  - benötigt längeren Kode und
  - mehr Schlüsselwörter als DINO
  - Entwickler muss auf Energiekonsum einzelner Aufgaben achten

Entwickleraufwand

- Alpaca
  - benötigt längeren Kode und
  - mehr Schlüsselwörter als DINO
  - Entwickler muss auf Energiekonsum einzelner Aufgaben achten
- Bei DINO muss Entwickler Kontrollpunkte setzen

Entwickleraufwand

- Alpaca
  - benötigt längeren Kode und
  - mehr Schlüsselwörter als DINO
  - Entwickler muss auf Energiekonsum einzelner Aufgaben achten
- Bei DINO muss Entwickler Kontrollpunkte setzen
  - Eigene Kodeanalyse notwendig

Entwickleraufwand

- Alpaca
  - benötigt längeren Kode und
  - mehr Schlüsselwörter als DINO
  - Entwickler muss auf Energiekonsum einzelner Aufgaben achten
- Bei DINO muss Entwickler Kontrollpunkte setzen
  - Eigene Kodeanalyse notwendig
- FlexiCheck arbeitet unsichtbar für Entwickler

Hardwareanforderungen

- DINO und Alpaca benötigen keine weitere Hardware

Hardwareanforderungen

- DINO und Alpaca benötigen keine weitere Hardware
- FlexiCheck benötigt spezielle Hardware
  - benötigt Mikrokontrollerunterstützung
  - Bei nicht optimaler Nutzung kann mehr Energie gebraucht werden [FVMH08]

Hardwareanforderungen

- DINO und Alpaca benötigen keine weitere Hardware
- FlexiCheck benötigt spezielle Hardware
  - benötigt Mikrokontrollerunterstützung
  - Bei nicht optimaler Nutzung kann mehr Energie gebraucht werden [FVMH08]
- DINO und Alpaca können auf bestehenden Plattformen betrieben werden

Vergleich

- DINO ist am langsamsten
  - Zusätzlich hoher Programmieraufwand
- Alpaca ist schneller
  - Ist duch Energievorrat auf Gerät limitiert
- FlexiCheck am schnellsten und kein Aufwand für Programmierer, **ABER**
  - kann nur auf neuen Plattformen verwendet werden
  - Hardwarekosten werden enorm erhöht
  - Aktuell keine kommerziellerhältliche Hardware
  - Hardware muss für jede neue Konfiguration trainiert werden

Fazit

Fazit

1. DINO allgemein sehr langsam

Fazit

1. DINO allgemein sehr langsam
2. FlexiCheck wesentlich schneller

Fazit

1. DINO allgemein sehr langsam
2. FlexiCheck wesentlich schneller
3. Bei bestehenden Plattformen Alpaca optimal

Fazit

1. DINO allgemein sehr langsam
2. FlexiCheck wesentlich schneller
3. Bei bestehenden Plattformen Alpaca optimal
4. Fraglich ob Hardwareentwicklung lohnt
    - Erhöht Gewicht und Volumen

Fazit

1. DINO allgemein sehr langsam
2. FlexiCheck wesentlich schneller
3. Bei bestehenden Plattformen Alpaca optimal
4. Fraglich ob Hardwareentwicklung lohnt
    - Erhöht Gewicht und Volumen
5. Neue Ansätze ohne Hardwareunterstützung evtl. schneller

Fragen

- Motivation
- Fehler
  - Datenzugrifffehler
  - Speicherabbild-Fehler
  - Fehler bei der Aktivierungsaufzeichnung
- Ansätze
  - DINO
  - Alpaca
- Vergleich
  - Laufzeit
  - Entwickleraufwand
  - Hardware
  - Ergebniss
- Fazit

Literatur (1)


C. Ferri, A. Viescas, T. Moreshet, and M. Herlihy, *Energyimplications of transactional memory for embedded architectures*, Workshop on Exploiting Parallelism with Transactional Memory andother Hardware Assisted Methods (EPHAM'08), April 2008.


Brandon Lucia and Benjamin Ransford, *A simpler, safer programming and execution model for intermittent systems*, Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation - PLDI 2015, ACM Press, 2015.


Kiwan Maeng, Alexei Colin, and Brandon Lucia, *Alpaca: intermittent execution without checkpoints*, Proceedings of the ACM on Programming Languages **1** (2017), no. OOPSLA, 1–30.